LIST

STORE A SEQUENCE OF DATA

- Each data can be of different type

For example:

```
data=['Introduction to Python',250.75,1005]
```

string float int

ACCESS

```
x=[0,1,5,4,10,14,16,20,2]
print(x[0])
print(x[3])
print(x[-1])
print(x[-2])
```

```
0
4
2
20
```

Slicing list_var[start:end:step]

```
x=[0,1,5,4,10,14,16,20,2]
x[:4]   # up to position just before position 4.
```

```
[0, 1, 5, 4]
```

```
x[2:]   # from position 2 onwards.
```

```
[5, 4, 10, 14, 16, 20, 2]
```

```python
x=[0,1,5,4,10,14,16,20,2]
```

```python
x[:10:2]    # from position 0 to 9, jump 2 data for each element.
```

```
[0, 5, 10, 16, 2]
```

```python
x[::2]      #from the first data to the last, jump 2 data at a time.
```

```
[0, 5, 10, 16, 2]
```

```python
x[::-1]     #reverse the list.
```

```
[2, 20, 16, 14, 10, 4, 5, 1, 0]
```

```
x[:]        #the same contents as the original.
```

```
[0, 1, 5, 4, 10, 14, 16, 20, 2]
```

```
x=x[:-1]    # This is x[0:-1:1] (not including the last data from the original).
```

change

```
x
```

```
[0, 1, 5, 4, 10, 14, 16, 20]
```

```
x=x[1:]    #x[1::1] (not including the first data).
```

```
x
```

```
[1, 5, 4, 10, 14, 16, 20]
```

```
In [36]: x

Out[36]: [1, 5, 4, 10, 14, 16, 20]

In [37]: x[-3:-1]    #x[-3:-1:1]

Out[37]: [14, 16]

In [38]: x[:100]    #x[0:100:1]

Out[38]: [1, 5, 4, 10, 14, 16, 20]
```

LEN()

```
x = [1, 5, 4, 10, 14, 16, 20]
len(x)
```

```
7
```

CONCAT

```
[1,2,3,4]+[6,2,1,'A']
```

```
[1, 2, 3, 4, 6, 2, 1, 'A']
```

```
[1]+[1]+[1]+[1]
```

```
[1, 1, 1, 1]
```

REPETITION

```
[1,2,3]*4
```

```
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]
```

INDEX(DATA,START_POSITION)

- Find the position of the given data (starting the search from position start_position)
- Default value for start_position is 0.
- If the data is not found, an exception is generated.

```
x = [1, 2, 3, 4, 6, 2, 1, 'A']
x.index(6)
```

```
4
```

```
x.index(1)
```

```
0
```

```
x = [1, 2, 3, 4, 6, 2, 1, 'A']
```

```
x.index(1,1)
```

```
6
```

```
x.index(7)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most rece
~\AppData\Local\Temp/ipykernel_26408/501634487.py in <module>
----> 1 x.index(7)

ValueError: 7 is not in list
```

```
if 7 in x:
  print(x.index(7))
else:
  print('Not found')
```

```
Not found
```

JOIN

- Str.join(list)
  - Produce a string by joining all list elements with Str.
  - List must be the list of strings. Otherwise, an error takes place!

```
tokens=['this','is','a','cat']
print(':'.join(tokens))
print(' '.join(tokens))
```

```
this:is:a:cat
this is a cat
```

ILLEGAL DATA FOR JOIN

```
data=[1,2,3,4,2]
' '.join(data)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most rece
~\AppData\Local\Temp/ipykernel_26408/1336469517.py in <module>
      1 data=[1,2,3,4,2]
----> 2 ' '.join(data)

TypeError: sequence item 0: expected str instance, int found
```

SPLIT

```
#     ' '.join(s.split())
s='this    is    a     rat'
l = s.split()
print(l)

s2 = ' '.join(l)
print(s2)
```

```
['this', 'is', 'a', 'rat']
this is a rat
```

this is a string

```
s='1,2,3,4,-1,5'
s1=s.split(',')
print(s1)
print(':'.join(s1))
```

```
['1', '2', '3', '4', '-1', '5']
1:2:3:4:-1:5
```

APPEND

- append(object)
  - Change the list by appending it with a given data.

```
x = [1, 2, 3, 4, 6, 2, 1, 'A']
x.append(100)     # x will CHANGE!!!!!
x  #print(x)
```

```
[1, 2, 3, 4, 6, 2, 1, 'A', 100]
```

INSERT

- insert(index, object)
  - Insert object at location index, other objects are moved to the right.

```
x = [1, 2, 3, 4, 6, 2, 1, 'A']
x.insert(3,200)   # x will CHANGE!!!!!
x
```

```
[1, 2, 3, 200, 4, 6, 2, 1, 'A']
```

```
x.insert(9,333)
x
```

```
[1, 2, 3, 200, 4, 6, 2, 1, 'A', 333]
```

```
x = [1, 2, 3, 200, 4, 6, 2, 1, 'A', 333]
```

```
x.insert(-1,444)
x
```

```
[1, 2, 3, 200, 4, 6, 2, 1, 'A', 444, 333]
```

```
x.insert(100,555)  #position outside does not cause error.
x
```

```
[1, 2, 3, 200, 4, 6, 2, 1, 'A', 444, 333, 555]
```

REMOVE VALUE

- remove(object)
  - Remove the first occurrence of object from the list (CHANGE the list!!!!)

```
x = [1, 2, 3, 200, 4]
x.remove(200)
x
```

```
[1, 2, 3, 4]
```

```
x.remove(999)
x
```

data does not exist.

```
---------------------------------------------------------------
ValueError                                Traceback (most rece
~\AppData\Local\Temp/ipykernel_26408/1397735924.py in <module>
----> 1 x.remove(999)
      2 x

ValueError: list.remove(x): x not in list
```

REMOVE POSITION

```
x = [1, 2, 3, 200, 4]
x.pop(1)
```

```
2
```

```
x
```

```
[1, 3, 200, 4]
```

```
x.pop(5)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recen
~\AppData\Local\Temp/ipykernel_26408/2007404377.py in <module>
----> 1 x.pop(5)

IndexError: pop index out of range
```

MODIFYING A LIST WITH SLICING

```
x = [1, 2, 3, 200, 4]
x[1:3]=['a','b','c']  #Replace a portion with a list (size may not be equal).
print(x)
```

```
[1, 'a', 'b', 'c', 200, 4]
```

```
x[1:1]=[1000,2000,3000]  # insert!!!
x
```

```
[1, 1000, 2000, 3000, 'a', 'b', 'c', 200, 4]
```

```
x[1:4]=[]
x
```

```
[1, 'a', 'b', 'c', 200, 4]
```

MODIFYING AN EXTENDED SLICE

```
[1, 'a', 'b', 'c', 200, 4]
```

```
x[::2]=[1000,2000,3000]
x
```

```
[1000, 'a', 2000, 'c', 3000, 4]
```

Must have equal number of data!!!!

```
x[::2]=[7,9,11,13]   #the modifier has more data than the sliced portion.
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_26408/2449327583.py in <module>
----> 1 x[::2]=[7,9,11,13]   #the modifier has more data than the sliced portion.

ValueError: attempt to assign sequence of size 4 to extended slice of size 3
```

LIST COMPREHENSION!!!

a list

data in the list

Where each data comes from

```
[i for i in range(10)]
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```python
[i*2 for i in range(10)]
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

```python
import random

[int(random.random()*100) for i in range(20)]
```

```
[31, 28, 90, 58, 42, 12, 17, 91, 41, 10, 95, 26, 28, 24, 39, 0, 83, 74, 75, 67]
```

LIST COMPREHENSION WITH CONDITIONAL TESTS

```
i for i in range(2,100) if i%2 != 0 and i%3 != 0 and i%5 != 0 and i%7 != 0 and i%11 != 0 and i%13 != 0]
```

```
[17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

```
def isOdd(i):
    return i%2 == 1


[i for i in range(1,20) if isOdd(i)]  # condition can be a function
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

LIST CONSTRUCTION

- Using assignment

```
x = [1000, 'a', 2000, 'c', 3000, 4]
y=x
```

```
x
```

```
[1000, 'a', 2000, 'c', 3000, 4]
```

```
y
```

```
[1000, 'a', 2000, 'c', 3000, 4]
```

What if x changes?

```
x[0]=10
```

```
x
```

```
[10, 'a', 2000, 'c', 3000, 4]
```

```
y
```

```
[10, 'a', 2000, 'c', 3000, 4]
```

Both change!

LIST CONSTRUCTION

- Using list()

```
x = [7, 'a', 2000, 'c', 3000, 4]
z=list(x)
```

```
z
```

```
[7, 'a', 2000, 'c', 3000, 4]
```

X and Z are separate copies.

```
x[0]=1000
```

```
x
```

```
[1000, 'a', 2000, 'c', 3000, 4]
```

```
y
```

```
[1000, 'a', 2000, 'c', 3000, 4]
```

```
z
```

```
[7, 'a', 2000, 'c', 3000, 4]
```

LIST CONSTRUCTION

- Reading lines from a file!

```
file=open('score.csv')
for line in file:
  print(line,end='')
file.close()
```

```
student_id,Q1,Q2,Q3,Q4,Q5
5600148421,7,1,6,6,6
5600163621,0,1,2,6,8
5600186321,6,5,9,10,3
5600334721,5,1,2,7,7
5600486621,3,9,9,7,4
5600555421,5,1,6,7,7
5600574721,3,3,4,8,4
5600612321,8,10,8,6,5
5600622121,5,9,6,10,7
```

```
file=open('score.csv')
lines=file.readlines()
file.close()
lines
```

```
['student_id,Q1,Q2,Q3,Q4,Q5\n',
 '5600148421,7,1,6,6,6\n',
 '5600163621,0,1,2,6,8\n',
 '5600186321,6,5,9,10,3\n',
 '5600334721,5,1,2,7,7\n',
 '5600486621,3,9,9,7,4\n',
 '5600555421,5,1,6,7,7\n',
 '5600574721,3,3,4,8,4\n',
 '5600612321,8,10,8,6,5\n',
 '560062212
```

To be continued….

Use list comprehension to create!

```
lines=[line.strip() for line in lines]
lines
```

```
['student_id,Q1,Q2,Q3,Q4,Q5',
 '5600148421,7,1,6,6,6',
 '5600163621,0,1,2,6,8',
 '5600186321,6,5,9,10,3',
 '5600334721,5,1,2,7,7',
 '5600486621,3,9,9,7,4',
 '5600555421,5,1,6,7,7',
 '5600574721,3,3,4,8,4',
 '5600612321,8,10,8,6,5',
 '5600622121,5,9,6,10,7',
```

```
lines[0].split(',')
```

```
['student_id', 'Q1', 'Q2', 'Q3', 'Q4', 'Q5']
```

```
IDs=[line.split(',')[0] for line in lines[1:]]
IDs
```

```
['5600148421',
 '5600163621',
 '5600186321',
 '5600334721',
 '5600486621',
 '5600555421',
 '5600574721',
 '5600612321',
 '5600622121',
```

HOW ABOUT GETTING THE SUM OF SCORES FROM THE GIVEN FILE

- Ok you need to know sum() first!

```
[1000, 'a', 2000, 'c', 3000, 4]
sum(x[::2])
```

```
6000
```

```
sum(x)
```

```
---------------------------------------------------------------
TypeError                                 Traceback (most rece
<ipython-input-30-714f1c07c870> in <module>
----> 1 sum(x)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

SEARCH STACK OVERFLOW

```
[26,
 17,
 33,
 22,
 32,
 26,
 22,
 37,
```

['student_id', 'Q1', 'Q2', 'Q3', 'Q4', 'Q5']

```
lines=[line.strip() for line in lines]
lines
```

```
['student_id,Q1,Q2,Q3,Q4,Q5',
 '5600148421,7,1,6,6,6',
 '5600163621,0,1,2,6,8',
 '5600186321,6,5,9,10,3',
 '5600334721,5,1,2,7,7',
 '5600486621,3,9,9,7,4',
 '5600555421,5,1,6,7,7',
 '5600574721,3,3,4,8,4',
 '5600612321,8,10,8,6,5',
 '5600622121,5,9,6,10,7',
```

```
numberLists = [line.split(',') for line in lines[1:]]
numberLists
```

```
[['5600148421', '7', '1', '6', '6', '6'],
 ['5600163621', '0', '1', '2', '6', '8'],
 ['5600186321', '6', '5', '9', '10', '3'],
 ['5600334721', '5', '1', '2', '7', '7'],
 ['5600486621', '3', '9', '9', '7', '4'],
 ['5600555421', '5', '1', '6', '7', '7'],
 ['5600574721', '3', '3', '4', '8', '4'],
```

```
n2 = [line[1:] for line in numberLists]
n2
```

```
[['7', '1', '6', '6', '6'],
 ['0', '1', '2', '6', '8'],
 ['6', '5', '9', '10', '3'],
 ['5', '1', '2', '7', '7'],
 ['3', '9', '9', '7', '4'],
```

```python
n2 = [line[1:] for line in numberLists]
n2
```

```
[['7', '1', '6', '6', '6'],
 ['0', '1', '2', '6', '8'],
 ['6', '5', '9', '10', '3'],
 ['5', '1', '2', '7', '7'],
 ['3', '9', '9', '7', '4'],
```

```python
n3 = [[int(i) for i in line] for line in n2]
n3
```

```
[[7, 1, 6, 6, 6],
 [0, 1, 2, 6, 8],
 [6, 5, 9, 10, 3],
 [5, 1, 2, 7, 7],
```

```python
n4 = [sum(line) for line in n3]
n4
```

```
[26,
 17,
 33,
 22,
 32,
 26
```

```python
scores=[sum([int(e) for e in line.split(',')[1:]]) for line in lines[1:]]
scores
```

```
[26,
 17,
 33,
 22,
 32,
 26,
```

GOING FURTHER (ADDING ID)

```
IDs=[line.split(',')[0] for line in lines[1:]]
IDs
```

```
['5600148421',
 '5600163621',
 '5600186321',
 '5600334721',
 '5600486621',
 '5600555421',
 '5600574721',
 '5600612321',
 '5600622121',
```

```
scores
```

```
[26,
 17,
 33,
 22,
 32,
 26,
```

```
[[IDs[i],scores[i]] for i in range(len(IDs))]
```

```
[['5600148421', 26],
 ['5600163621', 17],
 ['5600186321', 33],
 ['5600334721', 22],
 ['5600486621', 32],
 ['5600555421', 26],
 ['5600574721', 22],
```

DOING IT IN ONE LINE

```
[[line.split(',')[0],sum([int(e) for e in line.split(',')[1:]])] for line in lines[1:]]
```

```
[['5600148421', 26],
 ['5600163621', 17],
 ['5600186321', 33],
 ['5600334721', 22],
 ['5600486621', 32],
 ['5600555421', 26],
```

SORT AND SORTED

- list.sort() sort and change the original list.
- sorted(list) create a new list (sorted).

```
import random
data=[int(random.random()*100) for i in range(20)]
sorted(data)
```

```
[3, 7, 8, 9, 10, 17, 25, 37, 51, 53, 58, 64, 71, 71, 73, 77, 82, 87, 88, 93]
```

```
sorted(data,reverse=True)
```

```
[93, 88, 87, 82, 77, 73, 71, 71, 64, 58, 53, 51, 37, 25, 17, 10, 9, 8, 7, 3]
```

```
data
```

```
[58, 37, 87, 25, 9, 7, 82, 73, 8, 3, 71, 17, 53, 71, 88, 10, 77, 51, 64, 93]
```

```
[42] data.sort()
```

```
data
```

```
[3, 7, 8, 9, 10, 17, 25, 37, 51, 53, 58, 64, 71, 71, 73, 77, 82, 87, 88, 93]
```

SORTING COMPOSITE DATA

```
data=[[1,2],[1,3],[2,4],[5,7]]
sorted(data)
```

```
[[1, 2], [1, 3], [2, 4], [5, 7]]
```

```
data=[[1,2],[1,3],[2,4],[5,7]]
sorted(data,reverse=True)
```

```
[[5, 7], [2, 4], [1, 3], [1, 2]]
```

SORTING THE SUM LIST FROM PREVIOUS EXAMPLE

```
sorted([sum([int(e) for e in line.split(',')[1:]]) for line in lines[1:]],reverse=True)
```

```
[46,
 45,
 43,
 42,
 42,
 42,
 41,
 41,
 41,
 41,
```

```
sum_scores=[[line.split(',')[0],sum([int(e) for e in line.split(',')[1:]])] for line in lines[1:]]
sum_scores
```

```
[['5600148421', 26],
 ['5600163621', 17],
 ['5600186321', 33],
 ['5600334721', 22],
 ['5600486621', 32],
```

```
sorted([[element[1],element[0]] for element in sum_scores],reverse=True)
```

```
[[46, '5639110921'],
 [45, '5676237921'],
 [43, '5698907421'],
 [42, '5696614321'],
 [42, '5692820921'],
 [42, '5635209721'],
 [41, '5694033521'],
```

WRITE TO A NEW FILE (ADD COLUMN)

```
lines=[line.strip() for line in lines]
lines
```

```
['student_id,Q1,Q2,Q3,Q4,Q5',
 '5600148421,7,1,6,6,6',
 '5600163621,0,1,2,6,8',
 '5600186321,6,5,9,10,3',
 '5600334721,5,1,2,7,7',
 '5600486621,3,9,9,7,4',
```

'5600148421,7,1,6,6,6'

['7','1','6','6','6']

```
file=open('output.csv','w')
for line in lines[1:]:
  quiz_score=line.split(',')[1:]
  sum_of_score=sum([int(score) for score in quiz_score])
  print(line+','+str(sum_of_score))
  file.write(line+','+str(sum_of_score)+'\n')
file.close()
```

26

```
5600148421,7,1,6,6,6,26
5600163621,0,1,2,6,8,17
5600186321,6,5,9,10,3,33
5600334721,5,1,2,7,7,22
5600486621,3,9,9,7,4,32
```

write to file

'5600148421,7,1,6,6,6', '26'

SORT SCORE BY SUMMATION OF ALL QUIZZES

```
sum_score=sorted([ [sum([int(e) for e in line.split(',')[1:]]),line] for line in lines[1:]],reverse=True)
sum_score
```

```
[[46, '5639110921,10,10,8,10,8'],
 [45, '5676237921,10,10,9,9,7'],
 [43, '5698907421,10,10,6,10,7'],
 [42, '5696614321,5,10,9,10,8'],
 [42, '5692820921,7,10,8,9,8'],
```

```
file=open('sorted_score.csv','w')
for element in sum_score:
  file.write(element[1]+','+str(element[0])+'\n')
file.close()
```

```
! more sorted_score.csv
```

```
5639110921,10,10,8,10,8,46
5676237921,10,10,9,9,7,45
5698907421,10,10,6,10,7,43
5696614321,5,10,9,10,8,42
5692820921,7,10,8,9,8,42
```